Freescale Semiconductor
Application Note

Document Number: AN3289
Rev. 0, 07/2006

Low-Power Techniques for the S12X Family

by: Steve McAslan
MCD Applications, EKB

1 Introduction

Many applications depend on the ability of a microcontroller to provide high performance while using a minimum of power. The S12X family of microcontrollers provides a range of features to allow users to reach this goal. This application note examines those features, discusses how to combine them for maximum effect and identifies best practices when creating hardware.

2 Power Consumption in Different Operating Modes

In most embedded applications there are two distinct operating modes where power consumption is a key consideration. The first of these is stop mode, where the system is in a resting state but may require to examine activity in the system and respond quickly. The second of

Contents

these is run mode, where the system is fully operational and drawing its operational current from the power supply.

Two different characteristics of the microcontroller design affect the two different operating modes. For stop mode, the dominant characteristic may be the current drawn when the on-chip transistors are not switching. This is particularly the case when the MCU is not required to provide any functionality. In this case the current drawn is as a result of leakage through the CMOS gates on the device. Figure 1 illustrates the situation.

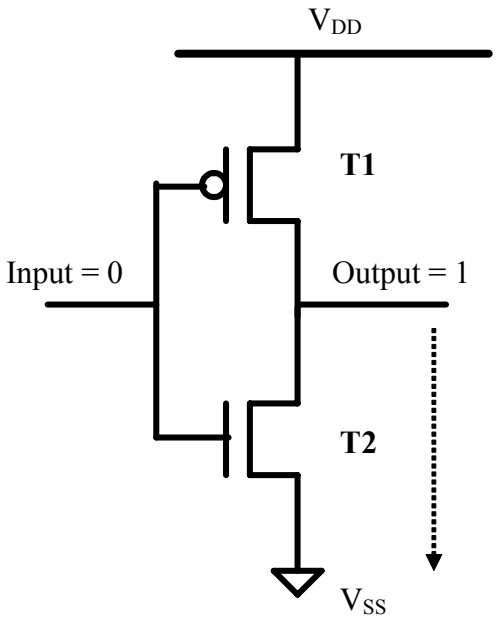

Figure 1. Leakage in a CMOS Gate

The CMOS transistors T1 and T2 are in a steady state where the input is not changing. In this configuration the input determines that T1 is in the on state and T2 is in the off state. Theoretically, no current can pass through T2; however, in practice a tiny current leaks through the gate. This is called the subthreshold or channel leakage current, and its value depends on the design of the transistor and in particular the operating voltage of the transistor. There are secondary effects that exist in this gate caused by parasitic diodes which are formed between the transistors and the substrate on which the transistors are formed. These diodes form additional leakage paths for current.

The leakage caused by these effects is minimized by design and production processes. Due to the nature of the leakage, however, the current taken is highly non-linear over temperature. At ambient or low temperature the stop current can be very small (in the order of 20–30 µA). However, at temperatures of 125 °C this can rise to much higher values - for example, typically 600 µA for the MC9S12XDP512 MCU.

In many applications the MCU has to offer some basic functionality while in stop mode, and this adds extra current due to the run mode effect. The extent to which it affects the functional stop mode depends on the amount of functionality in use.

The run mode current requirement also arises as a result of the design of the CMOS gate. Figure 2 illustrates the situation during run mode.

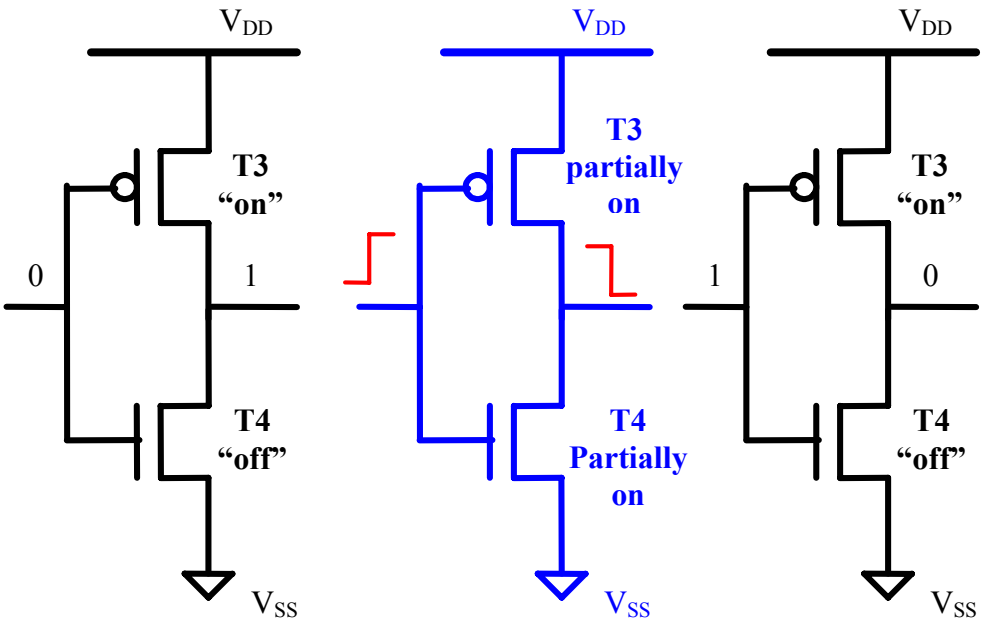

Figure 2. Run Mode Current in a CMOS Gate

In this mode, the power requirement is due to the switching effect of the gate. As the gate switches in response to a change on the input, the transistors T3 and T4 change from on to off (T3) or off to on (T4). This change in state takes a finite period of time (typically a few hundred picoseconds). During the transition, however, both transistors are conducting and so current can flow directly from the V_{DD} power line to the V_{SS} line. The more often the gate switches, the more current the gate consumes, and so a device like an MCU that operates from a common clock signal will see its current demand rise and fall depending on the clock frequency. This effect is largely linear, so a doubling of the clock frequency will typically result in a doubling of the power required.

In summary, the MCU power requirement depends on its operating mode and environmental conditions. In run mode, more current is drawn as the bus speed is increased; in stop mode, the dominant effect is likely to be temperature. This leads to the first suggested technique for reducing the power requirements on the S12X family (see item 1 in Table 1 at the end of this document).

3 Power Saving Modes of the S12X

The S12X MCU families provide chip-level controls for the performance of the device. These are Wait mode, Pseudo-stop mode and Stop mode. The characteristics of each of these are:

- Wait mode: stops the CPU completely, thus saving the power required to operate it. Other modules have variable levels of control, but typically have the option of continuing to operate or to stop. Since all of the clock chain continues to operate, the power savings are limited in this mode.

- Stop mode: this causes the crystal oscillator to stop and thus removes the primary clock from the device, effectively causing the MCU to stop all operations. This is the lowest power configuration of the device. Several modules on the MCU provide a way to restart the oscillator and restart code execution since in most cases applications still have a need to respond to system events. In addition, the S12X family MCUs include the option to wake the device after a time period has expired. This is a useful feature for systems that must periodically check the status of sensors within an application. The period is determined by a timer embedded in the on-chip voltage regulator called the Autonomous Periodical Interrupt (API). The API operates from an internal RC oscillator that can continue to operate when the crystal oscillator is stopped. Since this timer consists of logic switching due to a clock, it increases the total current drawn by the device. The total current is, however, still lower than in modes where the oscillator continues to run.
- Pseudo-stop mode: as the name suggests, this has many of the features of Stop mode. The difference is that the oscillator continues to operate with reduced amplitude and this allows some of the on-chip systems to continue to operate. In particular, the real time interrupt (RTI) and watchdog (COP) are available to provide precise timing references and protection for the system. Pseudo-stop mode also allows a much reduced time when restarting when compared to Stop mode due to the time taken by the crystal to start.

4 Operating Options in Wait Mode

The CPU enters this mode by executing the WAI opcode. This opcode prepares the CPU for a fast response to the event that will wake it by stacking its registers. The exact behavior of an MCU in wait mode depends on the configuration of the on-chip peripherals. In most cases the modules have the option of stopping immediately (or as soon as they complete the current operation) or continuing to operate. Examples of peripherals that stop immediately are the SCI, which will halt during the transmission or reception of a byte, and the timer, which will cease all timing operations. An example of a peripheral that will complete the current operation is the SPI, which will continue operating if configured as a slave.

Any general purpose I/O will retain its last setting (input or output) and thus, for example, any LEDs connected to the MCU will remain in their latest state and so consume current.

The behavior of each peripheral in Wait mode is controlled by a configuration bit. The minimum power configuration will be the one in which the most peripherals are powered down in Wait mode. See item 2 in Table 1 at the end of this document.

5 Power Saving by Varying Operating Frequency

As discussed, the current consumed by the MCU when operating is linearly related to the operating frequency. The MCU will therefore draw less current if it operates more slowly. Users can take advantage of this when operating from the internal PLL (IPLL) on S12XE family members.

Since the IPLL provides the bus clock for the MCU, and since this clock frequency can be changed at any time, it is possible for users to configure the operation speed of the MCU according to the processing demands of the application. If external communications are required during the lower speed mode then the baud rate selection for the communications modules may have to change.

The IPLL has additional flexibility in that it has a divider option on the clock output that allows the bus clock frequency to be lower than the external crystal's. The PLL frequency can be divided by up to 62, which gives a minimum MCU bus clock of approximately 260 kHz. See item number 3 in Table 1.

The IPLL is configured using the SYNR and REFDV registers to give a minimum operating frequency of 32 MHz. The POSTDIV register provides the clock output divider option.

6 Entering and Restarting from Stop Mode

The CPU enters this mode by executing the STOP opcode. This opcode prepares the CPU for a fast response to the event that will wake it by stacking its registers. The opcode will also cause most on-chip modules to cease their current operation. This can produce undesired effects and so it may be good practice to allow current operations to terminate before executing STOP. This is particularly the case with ongoing NVM operations. As in Wait mode, any general purpose I/O will retain its latest state and will continue to consume the current required by the external circuitry.

Execution of the STOP opcode is disabled at reset by the S-bit in the Condition Code Register (CCR). While this bit is set the CPU will ignore the STOP instruction and instead perform the equivalent of a two-cycle NOP opcode. The S-bit can be cleared using the various CCR-related opcodes. The STOP opcode is also ignored on the S12XE when the CPU is operating in User state.

Care should always be taken before entering Stop mode to ensure that the restart interrupt conditions are correctly configured.

Various events can cause the CPU to restart from Stop mode. These fall into three categories:

- Reset - the system asserts the RESET pin on the MCU, and the MCU restarts.
- XIRQ - this non-maskable interrupt will cause the MCU to restart in one of two ways. If the X-bit in the CCR is set, an XIRQ assertion will cause the CPU to restart from the opcode after the STOP opcode. In other words, the CPU does not jump to an interrupt service routine. If the X-bit is cleared, the CPU will restart at the XIRQ interrupt service routine.
- IRQ interrupt - this includes the external IRQ pin and the internal I-bit interrupt sources, (peripherals) and will cause the CPU to restart at the appropriate interrupt service routine. The I-bit must be cleared and the interrupt module configured appropriately before executing the STOP opcode to allow these interrupts to restart the MCU.

NOTE

The XIRQ input is level-sensitive and, if asserted, will cause the CPU to restart from STOP even if the XIRQ interrupt is disabled. Ensure that the hardware design takes this operating feature into account.

The exact configuration of the MCU in Stop mode depends on the state of system control bits when the STOP opcode is configured. For example, the difference between the Pseudo-stop mode and Stop mode is the setting of the PSTP bit.

When restarting from Stop mode S12X MCUs have two options:

- Start the crystal oscillator and run code once it has started. This can take several milliseconds due to the physical nature of the crystal oscillations.

- Start from the PLL VCO, which is an unregulated clock operating between approximately 1 MHz and 5 MHz. This can start in typically 50 µs.

The latter case is selected by setting the Fast Wake up (FSTWKP) bit which provides a low speed clock from which the MCU can operate for an unlimited time. The primary advantage of this selection is that the MCU can begin operating very rapidly after an external event occurs. The disadvantage is that the bus speed is unregulated and so cannot be used as a reference for communications peripherals. It is possible to enable the crystal oscillations at any time from the Fast Wake up condition. While the crystal is starting the MCU continues to execute code and when its start-up is complete the normal bus clock options are available.

7 A Note about Input Pins

As described previously, the status of the general purpose output pins is not affected by entry into Stop mode or Wait mode, and so they will continue to draw I/O current in these modes. (See item 4 in Table 1.) Most of these pins can also be set to input rather than output, and this introduces an additional responsibility for the hardware and software designer.

As shown in Figure 2, there is an input voltage on the transistors that causes both to be on simultaneously and this may occur for a short period while the gate is switching. The standard input port (or I/O port) on the S12X family has a similar design to this gate and consumes power in a similar way. The key difference is that the input voltage is not provided by an on-chip source but by an external voltage and so is not under the control of the MCU. Therefore it is possible to apply a mid-range voltage on the input and cause the gate to conduct current directly from the V_{DD} power line to the V_{SS} line. For this reason it is essential that only digital voltage levels are applied to any digital input pin.

An additional feature of the input pin is that the input impedance is very high. This can allow small charges on the input to accumulate and affect the behavior of the gate. In practice, there is a possibility for these charges to reach a steady state where the gate voltage is in the mid-range and the gate is again consuming current. For this reason all inputs must be externally or internally pulled into a known state when no external driving voltage is present. This also applies to the pins of the ATD converter when they are enabled as digital inputs.

The advice also applies to MCUs that are in packages that do not allow all I/O pins to be bonded out. In this case, the input pins are still present but they are not physically connected to any other hardware. These pins must be placed in a known state using either the internal pull devices or by changing them to an output. Failure to account for these floating input pins can result in hundreds of microamps of unwanted current consumption. See item 5 in Table 1.

8 Power Saving in Permanently Active Systems

In many applications the system is permanently powered up so that it can respond to external events. For most of the time the system must use as little power as possible, yet be able to become active if required. In this case the typical best practice is to place the MCU in one of the stop modes and configure it such that it can service a required event. The types of events that can cause the system to become active vary from application to application, but can be characterized as follows:

- Digital input events - where a logic level changes to indicate that the system must become active
- Analog events - where an analog level above or below a certain value means that the system must take action
- Communications events - where a message from another electronic module causes the application to wake up
- Periodic events - where the system must perform some activity at a known interval
- Random and complex events - where an event can occur randomly or where the application must perform some processing to determine if activity is required

The S12X family provides a variety of different features to address the above requirements; potential configurations for each are described below. Many systems will have to respond to a combination of the above events and will require a combination of configurations to allow this to occur.

8.1 Digital Input Events

Examples of this type of system event are a switch thrown by a user or a simple logic level change from an external module or sensor.

The sensor or switch could be connected to the dedicated IRQ or XIRQ pins if appropriate. Alternately, the S12XE and other S12X and S12 devices include extensive keyboard wake-up features designed specifically to detect this kind of event. Many general-purpose I/O pins on these devices have the option of creating an interrupt when the logical level on a pin changes. The ports typically include the option to trigger on active high or active low events and include pull-down or pull-up devices, respectively. For Port P on the S12XE family, the configuration registers are PERP to enable the individual pull-ups, PPSP to select the polarity of the edge detected, and PIEP to enable the interrupt when the edge occurs. Other ports have similar registers.

In this configuration, the MCU may be placed in Stop mode with no other functionality required and be restarted when the external logic change occurs.

8.2 Analog Input Events

This type of event occurs when a particular analog value is detected on an attached sensor. Examples here include temperature sensors and battery levels.

It is straightforward to design external circuitry to trigger on a particular analog event. This would most likely involve using an analog comparator to convert the analog event into a digital event and using the digital input event techniques to process the event.

On the S12XE family, a more sophisticated solution is possible using the on-chip ATD converter. This converter has two features that combine to allow detection of analog events. First, there is a digital comparator on each conversion channel that allows greater-than or less-than-or-equal-to comparisons. A positive comparison can cause the ATD to raise an interrupt. Second, the ATD module contains its own clock source in the form of an RC oscillator. This internal clock allows the ATD to make conversions even when the MCU is in Stop mode. By combining this self-clock mode with the comparator function, the user can configure the ATD to detect certain analog events and raise an interrupt that will restart the MCU from Stop.

The combination allows the MCU to restart when an analog voltage drops below a certain value or when it exceeds a value.

The internal clock mode of the ATD is enabled by the ICLKSTP bit, and each channel using the comparator is configured in the ATDCMPE register. The comparator function on each channel is determined by the status of bits in the ATDCMPHT register. Positive comparator results can raise an interrupt and restart the MCU if the ACMPIE bit is set.

8.3 Communications Events

In this scenario, the system is awoken by a message coming from another module. An example from the automotive industry would be messages on CAN or LIN networks that occur when the car has been parked and the driver is returning. The car locking mechanism would detect the unlocking of the doors and signal to the other in-car modules to wake over the network.

The S12X families allow both CAN and SCI (for LIN) ports to detect message events and raise an interrupt when one arrives. This feature allows the communications ports to restart the MCU from Stop mode. The user software must pre-configure the ports to allow this restart to occur.

The user software must place the MSCAN module in sleep mode with the wake-up enabled before the STOP opcode is executed. The CAN specification requires specific behaviors from compliant modules and unless the correct steps are followed the module will not be in the correct state for recovery. The process is described in detail in application note AN2255, *MSCAN Low Power Applications*.

The process is much more straightforward for the SCI module when used for LIN communications. After the LIN sleep command is issued the user software can set the Receive Input Active Edge Enable (RXEDGIE) bit to allow an active edge on the receive pin to restart the MCU.

8.4 Periodic Events

Often a system may have to restart on a regular basis. This may involve the maintenance of a real time clock or an external system that needs a regular check.

The appropriate configuration of the MCU in this case depends on the accuracy required for the periodic restart. If an accurate wake-up period is required (e.g., for real-time clock maintenance), the most appropriate low-power mode is Pseudo-stop mode. Since this mode keeps the crystal oscillator running, it provides an accurate timebase. The Real Time Interrupt uses the crystal clock as a source and can provide timeout periods from 250 µs to 800 ms from a 4 MHz crystal. The RTI has a choice of binary or decimal dividers, so there is a wide range intermediate values possible. A further benefit to Pseudo-stop mode is that the COP can also continue to run and so a failure in the operation of the periodic code can cause a watchdog recovery for the MCU. The primary disadvantage of Pseudo-stop mode is that the crystal oscillator consumes additional current so that the power requirements are higher than in Stop mode.

The PSTP and PRE bits enable Pseudo-stop and the RTI operation. The RTIE bit allows the RTI to interrupt and restart the MCU.

If a less accurate timeout is required, the API is a more energy-efficient solution. This operates with an accuracy of 10% over all operating conditions and uses approximately 10 times less current than Pseudo-stop mode with the RTI active.

Another operating requirement may involve a periodic review of the value of analog signals in the system. In this case it is possible to configure the ATD module to run from its internal clock, convert a series of channels, then raise an interrupt. At this point the MCU can restart and examine the conversion results. The ATD internal clock has a wide operating range so once again this is not an appropriate solution if accurate timing is required.

8.5 Random and Complex Events

In many cases it is not possible for the MCU to be directly stimulated by external activities and a more complex or proactive system examination is required. Examples of this kind include scanning a keyboard and checking the status of an external peripheral.

A key feature of this type of event is that most of the time the MCU will perform the status check and then go straight back to Stop mode. Only in a few cases will a restart cause the MCU to spend a significant time in run mode. Bearing this in mind, the most power-efficient configuration will be found by using the API to perform a periodic wake up and by selecting Fast Wake up mode. The combination allows the MCU to consume as little power as possible while in Stop mode, minimize its consumption when checking the status of the system and only activate full run mode if required. Figure 3 shows the flow of this type of application.

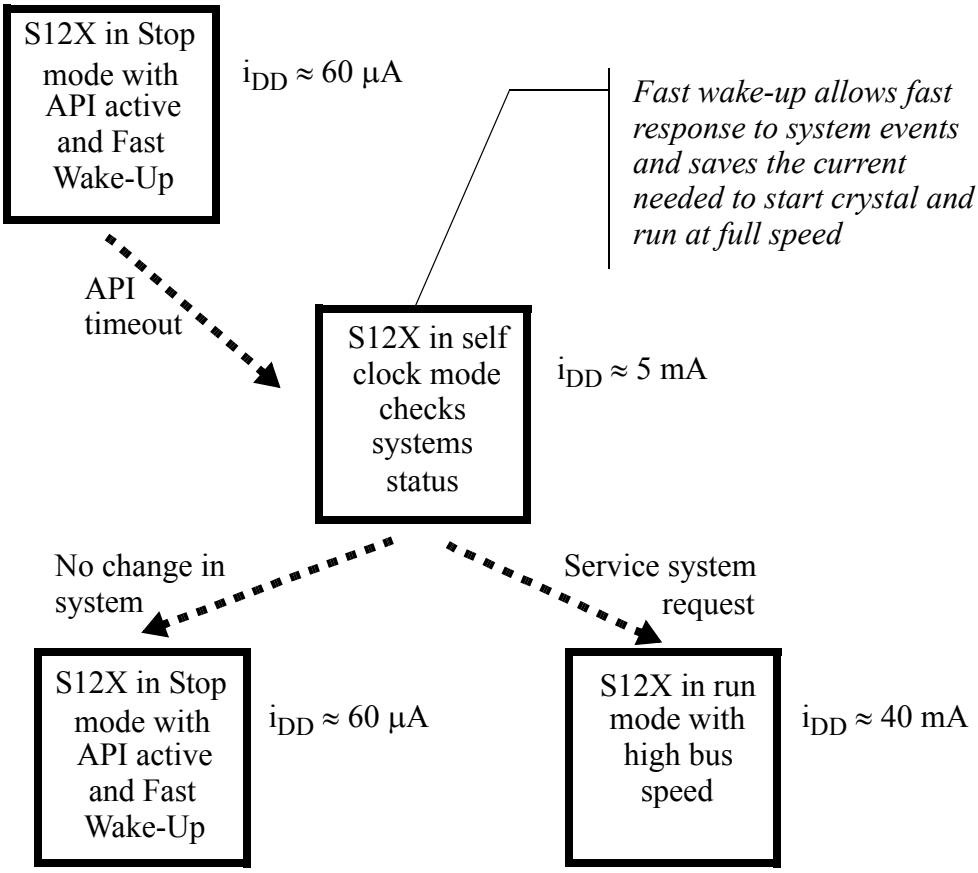

Figure 3. Example of Complex Event Handling

9 Using the Dual-Core Features of S12X to Save Power

The architecture of the S12X features the XGATE co-processor which, like the CPU, is an independently-programmable core. Unlike the CPU, the XGATE is a RISC processor that operates at twice the bus clock and consumes energy only when executing a thread to service an event. Unlike the CPU it has no need to specially select Stop mode when inactive.

The difference in the architecture between the CPU and XGATE opens the possibility of power savings since both take approximately the same amount of current when running. The faster core at a particular task is often the most power efficient solution.

For more information on the architecture of the S12X and the XGATE, please see application notes AN2685 and AN3144.

To see how the core architecture can influence the speed of execution (and therefore the power consumption) consider the C program shown in Figure 4. This is a very simple example of a periodic wake up that only enters run mode when some external event occurs but must also provide a changing indication to the system. The code is executed by an interrupt from the API which wakes the MCU from Stop mode with Fast Wake-up. It scans four I/O ports and compares the values against RAM, toggles an LED every eight API interrupts, and starts the crystal oscillator if the I/O value is different to the value in RAM.

By compiling the C code for both the XGATE and for the CPU, one can compare the relative execution times and hence power consumption. This comparison is shown in Figure 5, which shows that in this case the XGATE will complete the routine in less than half the time taken by the CPU. For this example it is therefore more efficient to use the XGATE to perform the low power processing.

```
Byte LoopCount,RAM_PortA,RAM_PortB,RAM_PortC;
Byte RAM_PortD,RAM_PortE;

// interrupt handler
interrupt void APIHandler(int dummy)
{
  Byte RAM_Changes = 0;
  if (PORTA != RAM_PortA) RAM_Changes |= 0x01;
  if (PORTB != RAM_PortB) RAM_Changes |= 0x02;
  if (PORTC != RAM_PortC) RAM_Changes |= 0x04;
  if (PORTD != RAM_PortD) RAM_Changes |= 0x08;

  LoopCount++;
  if (LoopCount%8) PORTE=PORTE^0x01;

  // Clear API interrupt
  VREGAPICL_APIF = 1;

  //Start up MCU if required
  if (RAM_Changes) PLLCTL_FSTWKP = 0;
}
```

Figure 4. C Example 1

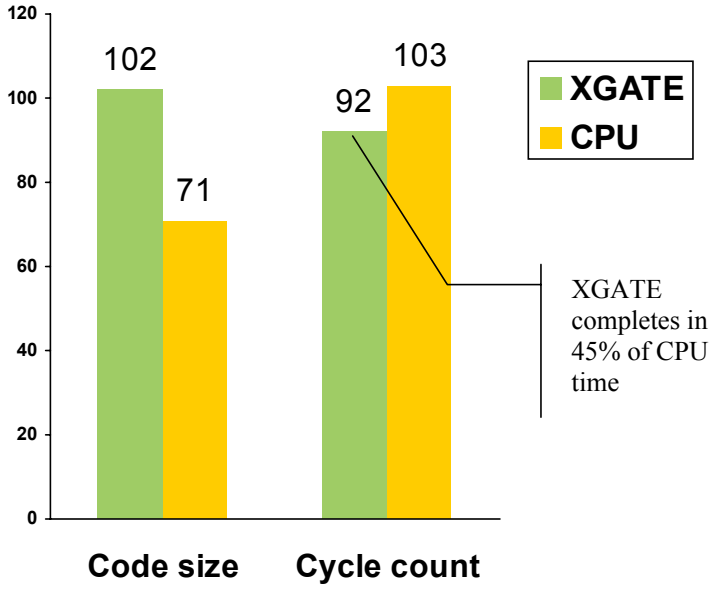

Figure 5. Example 1 Performance

Consider the second example, shown in Figure 6. In this case there are two LEDs, with one being toggled every 25 interrupts and one every 80 interrupts. The results for this example are shown in Figure 7. In this case the CPU is a better choice for the low-power processor.

```
Byte LoopCount;

// interrupt handler
interrupt void APIHandler(int dummy)
{
  LoopCount++;
  if (LoopCount%25) PORTE=PORTE^0x01;
  if (LoopCount%80)
  {
    PORTE=PORTE^0x02;
    //Start up MCU
    PLLCTL_FSTWKP = 0;
  }
  // Clear API interrupt
  VREGAPICL_APIF = 1;
}
```

Figure 6. C Example 2

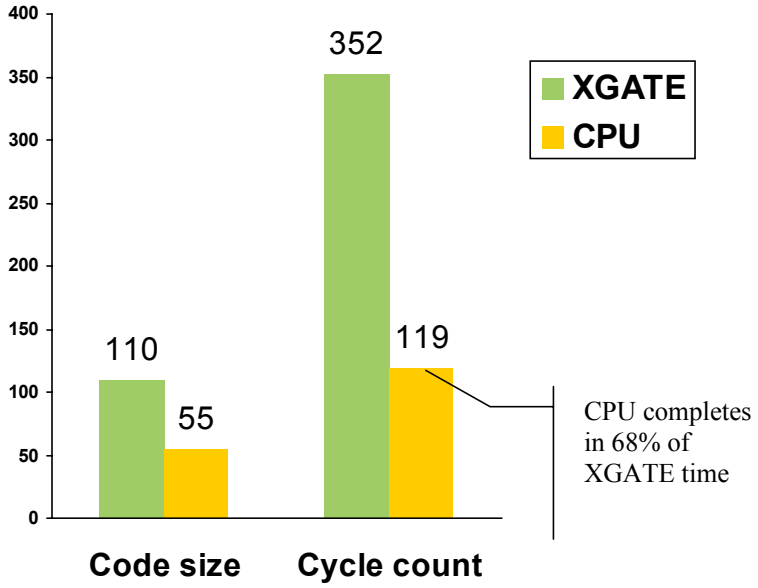

Figure 7. Example 2 Performance

In practice, the choice of the processor to use depends on the type of processing power required. XGATE is extremely efficient for operations involving bit manipulation, shifts, comparisons and RAM-based operations. The CPU is best when there is mathematical calculation required or extensive access to flash. Since both cores are easily programmed in C it is straightforward to write the algorithm and test its performance on each core in turn. (See item 6 in Table 1.)

10 Summary

The S12X and S12XE families provide a wide range of features to help reduce the operating power required. Table 1 summarizes six of the best techniques, all of which are described in this application note. A combination of these techniques may be required to take full advantage of the power-saving benefits of the MCUs.

Table 1. Power Saving Techniques

No.	Description	Comments
1	Reduce operating speed to maximum required for correct operation	Running faster than required consumes unnecessary power
2	Use the appropriate mode for application conditions	Select from wait, stop or pseudo-stop mode as required
3	Use the IPLL to vary the operating frequency	Reduce operating speed if less processing power is required
4	Set any I/O pins to a non current consuming state if possible	Current consumed by externally connected circuitry (LEDs etc.) will continue to be taken even in Stop mode
5	Terminate or avoid the use of floating inputs	Floating inputs can cause the input pin to consume unnecessary current
6	Choose the fastest core for a task when restarting from Stop mode	The availability of the XGATE and the CPU allows a choice of processor when an MCU restart is required

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3289
Rev. 0
07/2006

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

freescale™
semiconductor